Response to Final Office Action dated: September 26, 2006

Final Office Action dated: June 26, 2006

RECEIVED **CENTRAL FAX CENTER** SEP 2 6 2006

## AMENDMENTS TO THE CLAIMS

1-19 (Cancelled)

(Previously Presented) A method comprising: 20.

detecting a stall in an execution stage of a processor core;

generating a neutral instruction;

providing said neutral instruction to said execution stage; and

executing said neutral instruction to ascertain an architectural state value for said

processor core;

comparing said architectural state value for said processor core to an architectural state value for a second processor core.

- (Previously Presented) The method of claim 20 wherein said neutral instruction is 21. generated when a plurality of instructions are generated by a compiler.
- (Previously Presented) The method of claim 20 wherein said neutral instruction is 22. generated by a No-operation (NOP) pseudo-random generator.
- (Previously Presented) The method of claim 22 wherein the execution of said neutral 23. instruction causes said processor core to access a value stored in a register in said processor core.

Response to Final Office Action dated: September 26, 2006

Final Office Action dated: June 26, 2006

- 24. (Previously Presented) The method of claim 20 wherein the execution of said neutral instruction causes said processor core to access a value stored in a register in said processor core.
- 25. (Previously Presented) The method of claim 20 wherein said neutral instruction is generated by a post-processor device.
- 26. (Previously Presented) A system comprising:

stall logic coupled to an execution stage of a processor core to detect a stall in said execution; and

comparison logic coupled to said execution stage, wherein upon occurrence of the stall said execution stage is to execute a neutral instruction to ascertain an architectural state value for said processor core and compare said architectural state value with an architectural state value for a second processor core.

- 27. (Previously Presented) The system of claim 26 wherein said neutral instruction is generated by a compiler.
- 28. (Currently Amended) The system of claim 26 further comprising wherein:

a No-operation (NOP) pseudo-random generator coupled to the execution unit of said processor core to generate said neutral instruction.

said neutral instruction is generated by a No operation (NOP) pseudo-random generator.

Response to Final Office Action dated: September 26, 2006

Final Office Action dated: June 26, 2006

- 29. (Previously Presented) The system of claim 28 wherein the processor core includes a register and the execution of said neutral instruction causes said processor core to access a value stored in the register in said processor core.
- 30. (Previously Presented) The system of claim 29 wherein said neutral instruction includes
  ORing the contents of said register with itself.
- 31. (Previously Presented) The system of claim 29 wherein said neutral instruction includes ANDing the contents of said register with all binary 1 values.
- 32. (Previously Presented) The system of claim 29 wherein said neutral instruction includes ORing the contents of said register with all binary 0 values.
- 33. (Previously Presented) A set of instructions residing in a storage medium, said set of instructions capable of being executed in an execution stage by a processor core for implementing a method to test the processor core, the method comprising:

detecting a stall in an execution stage of a processor core;

generating a neutral instruction;

providing said neutral instruction to said execution stage;

executing said neutral instruction to ascertain an architectural state value for said processor core; and

Response to Final Office Action dated: September 26, 2006

Final Office Action dated: June 26, 2006

comparing said architectural state value to an architectural state value for a second processor core.

- 34. (Previously Presented) The set of instructions of claim 33 wherein in said method said neutral instruction is generated when a plurality of instructions are generated by a compiler.
- 35. (Previously Presented) The set of instructions of claim 33 wherein in said method said neutral instruction is generated by a No-operation (NOP) pseudo-random generator.
- 36. (Previously Presented) The set of instructions of claim 35 wherein in said method the execution of said neutral instruction causes said processor core to access a value stored in a register in said processor core.
- 37. (Previously Presented) The set of instructions of claim 33 wherein in said method the execution of said neutral instruction causes said processor core to access a value stored in a register in said processor core.
- 38. (Previously Presented) The set of instructions of claim 33 wherein in said method said neutral instruction is generated by a post-processor device.